Heterogeneous Media Packet Bridging

Copyright Notice/Permission

Field of the Invention

[0002] The present invention is related to media packet bridging, and more particularly to bridging heterogeneous media packets received from network interfaces from a processing resource.

Background Information

[0003] Ethernet Local Area Networks (LANs) are used to connect multiple computing devices to a network. Generally a LAN configuration is sufficient when the number of computing devices (e.g., computers, appliances, peripherals) communication with one another is relatively small. However, when the number of computing devices increase, multiple networks or network segments will require interconnection. There are a variety of techniques for connecting multiple network segments. One of the easiest and oldest techniques for connecting multiple network segments is using an Ethernet Bridge. Ethernet bridges have multiple Ethernet Network Interfaces, which connect to networks that require a bridge.

[0004] Apart from being able to interconnect Ethernet Networks, bridges can also help with segregating network traffic. For example, in an enterprise with 5 departments having 25 computing devices each, on simple way of connecting all 25 computing devices to a single LAN. While this technique will work, the technique has some obvious disadvantages, since computing devices associated with one department of the enterprise will be disrupted when two computing devices from another department communicate with one another. Thus, a more efficient network design for enterprise is to have a separate LAN for each separate department, where each separate LAN is interconnected with a bridge. In this way, all intra-department traffic will not be present to disrupt other traffic. Moreover, only inter-department traffic will be present on the bridge.

[0005] As previously discussed, Ethernet bridges are one of the oldest techniques for interconnecting LANs. Since the inception of Ethernet bridges, they have been enhanced to serve a variety of needs. Once such need relates to connecting (e.g., bridging) multiple LANs that are not in geographic proximity to one another. In these circumstances, Ethernet is transmitted on top of a different framing media (e.g., Asynchronous Transfer Mode (ATM), Gigabit Ethernet (GigE), Frame Relay (FR), Time-Division Multiplexing (TDM), and others). This creates an enhanced bridge with different framing media to external networks.

[0006] The enhanced bridge is capable of peeling off the framing headers to detect the Ethernet packet and then performing standard Ethernet bridging operations as if the data were received from a standard LAN interface. This type of enhanced Ethernet Bridge is widely implemented in enterprises with large network branches that are geographically dispersed in order to interconnect the LANs associated with the network branches.

[0007] There are a number of conventional products that perform Ethernet bridging over Ethernet media, Ethernet bridging over FR media, Ethernet

bridging over ATM media, and the like. However, these conventional products do not permit media agnostic Ethernet bridging. In other words, conventional approaches use a separate and often hardwired network resource that communicates with a specific network interface media. Each network resource is dedicated to a network interface in order to provide Ethernet bridging for a specific media type (e.g. ATM, GigE, FR, TDM, and others) handled by the network interface.

[0008] By dedicating network resources to specific network interfaces, an enterprise's heterogeneous networks are not efficiently and flexibly interconnected. Many times the dedicated resources are the result of an enterprise gradually growing its networks, with a later decision to bring the networks together. Alternatively, enterprises can merge previously disconnected departmental networks, or merge with other enterprise networks, and there exist a desire to interconnect the heterogeneous networks. Yet, with conventional approaches the bridging domains for each disparate network is isolated using separate network resources. As one of ordinary skill in the art appreciates, this can be expensive, inflexible, and time consuming for an enterprise to implement.

[0009] Therefore, there exist a need for techniques that provide improved heterogeneous network bridging, which are media agnostic. Thus, network resources need not be hardwired or dedicated to a single network interface, and a single bridging domain can be used to bridge all media transmissions.

Summary of the Invention

[0010] According to one aspect of the present invention, a method to bridge network packet media is presented. A first network packet is received from a first media channel via a first network interface. Also, a second network packet is received from a second media channel via a second network interface. The first network packet and the second network packet are

relayed to a single processing resource application. The application bridges the first network packet to the second media channel via the second network interface and the second network packet to the first media channel via the first network interface.

[0011] According to another aspect of the present invention, a network packet media bridging system is provided. The bridging system includes a plurality of network interfaces and a bridging application. Each network interface accepts network packets from a media transmission channel. The bridging application is accessible to a single processing resource and receives the network packets from the network interfaces. Further, the bridging application translates a number of the network packets between media formats for delivery to a number of the media transmission channels.

Brief Description of the Drawings

[0013] Fig. 1 shows a diagram of a network packet media bridging system, according to the present invention;

[0014] Fig. 2 shows a flow diagram of a method for bridging network packet media, according to the present invention;

[0015] Fig. 3 shows a flow diagram of another method for bridging network packet media, according to the present invention; and

[0016] Fig. 4 shows a diagram of another network packet media bridging system, according to the present invention.

Description of the Preferred Embodiments

[0017] In the following detailed description of various embodiments of the present invention, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood

that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

[0018] As used herein, a "network interface" or a "network media interface" (netmods) is a hardware and software-computing device that connects to telecommunications lines associated with network feeds. Netmods are well known to one of ordinary skill in the art. Netmods come in a variety of configurations and are usually distinguished by the number of telecommunication lines that can physically connect to line interface ports of the netmod.

[0019] Netmods include firmware and software to process raw packet data being received on a line interface port. Furthermore, some software instructions are processed within a volatile memory of the netmods. For example, some software instructions permit the recognition of separation of network data packets from a data stream being received over a line interface port. Additionally, software instructions can assign and regulate priorities to data packets being sent from the netmods back over a line interface port.

[0020] In various embodiments of the present invention, conventional netmods are used to achieve the teachings of the present invention. The netmods are also connected on the backend (e.g., the side opposite the network feed) to a switching fabric that is used to forward a network packet received from the netmod to one or more processing resources. The processing resources include one or more processing elements and memory. Additionally, the processing resources include applications that are used to translate, encrypt/decrypt, authenticate, forward and/or route any network packets received from the switching fabric.

[0021] In one embodiment of the present invention, a plurality of netmods, a switching fabric, and a plurality of processing resources are assembled as a network routing/switching device, such as a blade server. The blade server is configured and distributed by Cosine Communications, Inc. of Redwood

City, California. The blade server can be assembled with a plurality of additional blade servers that interface with one another. Moreover, IPNOS product offerings, distributed by Cosine Communications, Inc. of Redwood City, California can be modified to embody the teachings of the present disclosure. Of course as one of ordinary skill in the art readily appreciates, any hardware, firmware, and/or software configurations/products that are designed to achieve the tenets of the present disclosure can be used. Thus, all such configurations/products are intended to fall within the scope of the present invention.

[0022] Fig. 1 illustrates a diagram of a network packet media bridging system 100, according to the present invention. The bridging system 100 includes a plurality of netmods (e.g., 110 and 120), a switching fabric 112, and a processing resource 130. The processing resource 130 includes a bridging application 132 executing on or accessible to the processing resource 130. The netmods (e.g., 110 and 120) are connected to telecommunication lines associated with other networks (e.g., 140 and 150). Connections to the telecommunications lines are made via line interface ports included within the netmods (e.g., 110 and 120).

[0023] The netmods (e.g., 110 and 120) include memory and processing elements for receiving network packets from the line interface ports or for sending network packets out over the line interface ports. In some cases, the memory included within the netmods (e.g., 110 and 120) is Static Random Access Memory (SRAM), which is volatile memory permitting fast access to data. Moreover, the netmods (e.g., 110 and 120) are associated with a specific type of media transmission channel (e.g., ATM, GigE, TDM, FR, and the like). Additionally, a netmod (e.g., 110 or 120) can be wireless. Thus, network netmods (e.g., 110 and 120) need not be physically connected to a telecommunications line, but, rather, can be a transceiver for

transmitting and receiving wireless (e.g., Radio Frequency (RF), Infrared (IR), Satellite, and the like) network packets.

[0024] The switching fabric 112 is hardware, firmware, and, in some instances, software instructions that receive forwarded network data packets from the netmods (e.g., 110 and 120) and rapidly transfer the packet to the processing resource 130. Conventionally, switching fabric is hardwired from a specific netmod to a specific processing resource for each disparate media transmission (e.g., ATM, GigE, TDM, FR, wireless, and the like). The switching fabric 112 can also receive network packets from the processing resource 130 and forward the network packets along to the appropriate netmod (e.g., 110 and 120).

[0025] A number of applications executing on the processing resource 130 receives network packets and performs a variety of translations/operations on the network packets, such as forwarding, routing, encryption/decryption, authentication, and the like. Additional applications executing on the processing resource 130 can also be used to communicate with other processing resources (not shown in Fig. 1).

[0026] The processing resource 130 includes a bridging application 132. The bridging application 132 translates network packets from disparate media formats received from disparate netmods (e.g., 110 and/or 120). Each network packet received by the bridging application 132 includes metadata associated with the addressing particulars of each network packet. The metadata also includes Ethernet header data that is transmitted on top of the network data packets over the disparate media transmission channels.

[0027] The bridging application 132 inspects this metadata to strip the Ethernet header. The Ethernet header data allows the bridging application to associate the network packets with a standard intermediate media format. Moreover, once the network packets are associated with the intermediate format, and the next address locations for the network packets determined,

then the bridging application 132 translates the network packets into a media format required of any destination (e.g., next address location) netmod (e.g., 110 and 120).

[0028] If the received network packet is in a first media format and is destined to be relayed to a netmod (e.g., 110 or 120) associated with a second media format, then the bridging application 132 uses the metadata information available for each media format and translates the received network packet to the second media format before relaying the network packet along to the appropriate netmod (e.g., 110 or 120) that is handling network traffic associated with the second media format. This is achieved by using the Ethernet header data to first translate the received network packet to Ethernet and then to the second media format.

[0029] In one embodiment, the bridging application 132 has access to a translation table that permits it to translate from an Ethernet media format to the desired media formats. The table includes row identifiers for metadata elements and column identifiers for the media formats. Each cell of the table includes the corresponding equivalent for a particular metadata element and a particular media format. For example, a metadata header element can be identified by indexing to a row in the table having an identifier associated with headers, such as "r-x," where r identifies a row entry and x identifies a specific metadata element. Moreover, the desired translation from Ethernet format to a desired second media format can be acquired by indexing the appropriate column "c-f" for the desired media format, where c represents a column entry and f is the desired media format.

[030] For example if the originally received media format is FR and the desired media format is ATM, then the FR format is first translated to Ethernet using the Ethernet header transmitted on top of the packet in FR format. As one of ordinary skill in the art appreciates, a Frame Relay Access Device (FRAD) can be used to achieve the transmission of Ethernet on top

of a FR transmission. Next, a single element of the Ethernet intermediate format is identified as a specific row entry "r-01" within the table, and by accessing the column identifier "c-ATM" associated the desired ATM format; a table cell is constructed by the tuple "r-01, c-ATM". The tuple provides the appropriate translation information to the bridging application 132 for translating the intermediate Ethernet format to ATM.

[031] Thus, the bridging application 132 translates a network data packet from FR to ATM. Of course a variety of more complex translation can be required, such that the cells of the table process other applications in order to complete the translation from FR to ATM. All such translations are intended to fall within the scope of the present disclosure. Further, it is readily apparent that the bridging application 132 does not require a table for translations, since the logic to perform the translations can be embodied within the bridging application 132.

[0031] Thus, in some embodiments, the bridging application 132 can access a more complex table in order to perform the appropriate translations on the media formats, such as when metadata elements of one media format does not map directly (e.g., one to one relationship) to a like metadata element in another media format. In these cases, the cells of the table can include the appropriate translation instructions (e.g., pointers to other applications) or mappings, and the bridging application 132 is adapted to process or initiate these instructions or mappings.

[0032] Of course, the bridging application 132 need not include tables at all rather it can access software applications or software objects that assist in performing the appropriate media format translations. Moreover, in some cases, the bridging application 132 can access a plurality of translation tables each linked to one another for purposes of performing the appropriate translations between media formats. Thus, the bridging application need not, in all circumstances, first translate a received network packet to an

intermediate Ethernet format. For example, parsing requirements for separating the different elements associated with the metadata of a particular media format can be initially acquired by indexing on the received media data format in order to acquire the appropriate parsing instructions for the received metadata. Furthermore, as one of ordinary skill in the art readily appreciates, the translation information included within any cell of a table can be a direct mapping or a pointer to another application that knows how to perform the appropriate translation to a desired format. In this way, the translation information can be more complex and used to execute additional applications. And, in some cases, the additional applications can reside on additional and separate processing resources from the one in which the bridging application 132 is executing.

[0033] Furthermore, in some embodiments, the bridging application 132 can be instantiated, designed, and/or configured from a Graphical User Interface (GUI) application interfaced to the processing resource 130. Thus, as more translations between disparate media formats are desired, the bridging application 132 can be configured to accommodate the translation. Moreover, the configuration can be dynamic, so that the bridging application 132 need not be recompiled and re-linked when modifications are made. This can be achieved by using dynamic objects that are dynamically bound to the bridging application 132, or in the cases where the bridging application 132 acquires translation information from a dynamically modifiable table.

[0034] Accordingly, network packets associated with disparate media formats are received from different netmods and relayed to a single processing resource 130 having access to a bridging application 132. The bridging application 132 provides a virtual bridge between the disparate media formats, by translating the media formats of the network packets, as needed, before relaying any translated network packet to an appropriate netmod (e.g., 110 or 120).

[0035] As is now apparent to one of ordinary skill in the art, the present embodiments of the bridging system 100 offers significant improvements over conventional techniques, since a single processing resource 130 can execute a bridging application 132, which acts as a single bridging domain for all network packets. The virtual bridging system 100 is, therefore, not hardwired (as what has been conventionally required), and the virtual bridging system 100 can be dynamically configured and modified to adjust to the changing network patterns and needs of a network.

[0036] Fig. 2 illustrates a flow diagram of a method 200 for bridging network packet media, according to the present invention. In one embodiment, of Fig. 2 the method 200 is implemented within a high-density server or blade server having a plurality of netmods, a switching fabric, and a plurality of processing resources. In other embodiments, the method 200 is implemented within any network router, network switch, or network-computing device. Each processing resource, can receive network packets from a number of the netmods, where each netmod is associated with a different media transmission channel and media format (e.g., ATM, GigE, TDM, FR, wireless, and the like). Of course, any configuration of computing devices implementing method 200 is intended to fall within the scope of the present disclosure.

[0037] In 210, a first netmod receives a first network packet from a first media channel, and, in 220, a second netmod receives a second network packet from a second media channel. Each network packet is in a data format associated with its respective media channel. The first and second netmods are designed to relay or steer the received network packets to a single processing resource, as depicted in 230. In some embodiments, the first and second packets also include Ethernet transmitted on top of its native media data format.

[0038] The processing resource includes a bridging application, which is processed when the packets are received by the processing resource. The bridging application can execute one or more instructions and/or access one or more tables (or any data structures) in order to bridge the first network packet to the second media channel, as depicted in 240. Additionally, the bridging application can execute one or more instructions and/or access one or more tables to bridge the second network packet to the first media transmission channel, as depicted in 250.

[0039] In some embodiments, and in 260, the bridging application inspects the network packets for addressing information or metadata information. The information associated with any received network packet is inspected to determine where the received network packet is to be relayed next. If the relay location is a netmod associated with a disparate media format and a disparate media channel from what the received network packet was originally received in, then the bridging application translates the received network packet to an intermediate Ethernet format by using the network packet's Ethernet header included with the network packet. The intermediate Ethernet format is then used to translate the network packet to a destination media format before relaying the network packet to the appropriate netmod. In some cases, this translation can entail converting only the metadata information associated with the network packet.

[0040] In some instances, the bridging application can determine, upon inspecting a received network packet, that to properly translate the received network packet, one or more transfers need to occur by relaying the network packet to a second application residing on a second processing resource, as depicted in 270. Alternatively, the bridging application can relay the received network packet to a second application executing on the same processing resource as the bridging application.

[0041] The bridging application is a virtual bridge between heterogeneous network packet media. In some embodiments, the bridging application can be dynamically instantiated, configured, and/or modified by using a GUI application interfaced to the processing resource. Thus, method 200 provides a virtual bridge from within a single processing resource to translate between disparate media formats and disparate media channels. This virtual bridge provides a single bridging domain for network packets of disparate media formats.

[0042] Fig. 3 illustrates a flow diagram of another method 300 to bridge network packet media, according to the present invention. In some embodiments, the method 300 is implemented within a high-density or blade server. The blade server includes a plurality of network interfaces, a switching fabric, and a plurality of processing resources. Additionally, the method 300 can be implemented within any network router, network switch, or network-computing device. However, any configuration of computing devices implementing method 300 is intended to fall within the scope of the present disclosure.

[0045] In 310, a first network packet is received and is identified as being associated with a first media format (e.g., GigE, ATM, FR, TDM, wireless, or others). Additionally, in 320, a second network packet is received and is identified with a second media format. In some embodiments, the network packets are received from disparate netmods where each netmod is designed to relay and transmit network packets from and to a specific media transmission channel (e.g., GigE, ATM, FR, TDM, wireless, or others).

[0046] In 330, a translation data structure is accessed to translate the second network packet from the second media format to the first media format. Additionally, in some embodiments, and in 340, the translation data structure is accessed to translate the first network packet from the first media format to the second media format. In this way, the translation data structure acts as

a virtual bridge between disparate media formats associated with the network packets. Further, in one embodiment, the first and second network packets are translated to Ethernet format before translation to a desired media format occurs. This can occur, when the network packets also include Ethernet transmitted on top of their native media formats.

[0047] In some embodiments, the translation data structure is configurable and instantiated within a single processing resource by using a GUI application interfaced to the processing resource and the translation data structure, as depicted in 350. The translation data structure can be one or more data structures (e.g., tables, lists, trees, and the like) logically linked together. Moreover, information within the translation data structure can permit the execution of instructions (e.g., pointers to other external applications) in order to assist with translation between heterogeneous media formats.

[0048] Moreover, in some cases, the translation data structure can be implemented within a bridging application that executes on a single processing resource, where the processing resource receives the network packets from a plurality of netmods associated with disparate or different media transmission channels and formats.

[0049] Once, the translation data structure is accessed to translate or bridge the disparate media formats, then, in 360, the translated network packet is provided to the appropriate netmod associated with the translated media format. Thus, the translation data structure is used as a virtual bridge between heterogeneous packet media. The translation data structure is dynamically configurable and modifiable to accommodate a plurality of media translations desired within a network.

[0050] Also, as one of ordinary skill in the art will readily appreciate, translation data structure is used on Layer 3 (e.g., network or IP layer), and is capable of linking the necessary metadata associated with addressing various

media formats. Metadata includes header information and other information used in resolving network packet addresses. Disparate media formats can include disparate metadata. Conventional approaches resolve the disparate addressing between media formats by dedicating a single processing resource having applications that are used to translate the metadata for each media format. In the present invention, a single processing resource utilizes application(s) that accesses the translation data structure for received and translated network packets, in order to bridge heterogeneous packet media. Thus, in various embodiments of the present invention, a single bridging domain is provided for media bridging.

[0051] Fig. 4 illustrates a diagram of another network packet media bridging system 400, according to the present invention. The media bridging system 400 includes a plurality of netmods (e.g., 410 and 420), a relaying data structure (e.g., 412 and 422) for each netmod (e.g., 410 and 420), and a bridging application 432 that resides in a processing resource 430. The bridging system 400 can be implemented in a high-density or blade server. Alternatively, the bridging system 400 can be implemented in any network router, network switch, of network computing device. Of course, other configurations of computing devices that provide the media bridging of the present invention can also be implemented, without departing from the present invention.

[0052] Each netmod (e.g., 410 and 420) includes a plurality of line interface ports (not depicted in Fig. 4). The line interface ports accept network traffic from a telecommunications line (a transceiver when the network traffic is wireless). The netmods (e.g., 410 and 420) identify and select network packets from the network traffic occurring on the line interface ports. Each of the netmods (e.g., 410 and 420) is associated with different media channels (e.g., GigE, ATM, TDM, FR, wireless, and others).

[0053] The relaying data structures (e.g., 412 and 422) are accessed when network packets are identified by the netmods (e.g., 410 and 420). The relaying data structures (e.g., 412 and 422) permit the netmods (e.g., 410 and 420) to relay the network packets to the processing resource 430. The relaying data structures (e.g., 412 and 422) are dynamically configurable within the processing resource 430 and provided to the netmods (e.g., 410 and 420). In some embodiments, the relaying data structures (e.g., 412 and 422) are represented as SRAM tables within the netmods (e.g., 410 and 420). The SRAM tables (e.g., 412 and 422) can include identifiers for the processing resource 430, the netmods (e.g., 410 and 420), and identifiers for line interface ports (not depicted in Fig. 4).

[0054] When the network packets are relayed from the netmods (e.g., 410 and 420), they can include an Ethernet transmitted on top of the native media format. However, the metadata-addressing format of the network packets may still be in a format associated with the original media channel format.

[0055] The bridging application 432 receives the relayed network packets and detects the original media channel formats for the network packets based on metadata associated with the network packets. The bridging application 432 then translates a number of the network packets from a received media channel format to a requisite media channel format, based on where a particular network packet is to be relayed to next. Yet, as one of ordinary skill in the art readily recognizes, this translation can be done on an Ethernet format, when the network packets include Ethernet on top of their native media formats. Finally, the bridging application 432 translates any number of the network packets from the received media channel formats to requisite media channel formats. In this way, the bridging application 432 uses metadata associated with disparate media formats to bridge the network packets between heterogeneous media formats using traditional Ethernet, or

any other intermediate media channel format transmitted on top of the network packets along with their native media formats.

[0056] In one embodiment, the bridging application 432 communicates with one or more additional processing resources (e.g., 440 and 450). The bridging application 432 can use these additional processing resources (e.g., 440 and 450) to assist in bridging between heterogeneous media formats. In this way, processing can be offloaded from the processing resource 430 to the additional processing resources (e.g., 440 and 450).

[0057] In some embodiments, the bridging application 432 is dynamically instantiated and configured through a GUI application communicating with the processing resource 430. The configurations can include parameters that identify the media formats that the bridging application 432 is capable and permissibly allowed to bridge. The bridging application 432 can be provided as an Application Programming Interface (API) library, or as an OO class object having public and private methods. The API can be provided as a Dynamic Linked Library (DLL) or a shared library. Of course, any implementation, including stand alone ad hoc implementations, of the bridging application 432 that is designed to bridge heterogeneous media formats from a single processing resource 430 is intended to fall within the scope of the present invention. Moreover, the bridging application 432 can use one or more tables or other data structures to bridge heterogeneous media formats.

Conclusion

[0058] Methods and systems detailed above packet media bridging in a network. These methods and systems create a single media bridge domain for use in network routing environments. In contrast, traditional approaches have relied on hardwired and static implementations of switches and media bridges, thereby creating a plurality of media bridging domains. Accordingly, the present invention permits better utilization and load balancing of an enterprise's network routing resources.

[0059] Furthermore, the virtual media bridges of the present invention are dynamically configurable to meet the changing needs of an enterprise's network traffic. In some embodiments, the configuration of the virtual bridges can be altered using a GUI application in communication with a processing resource. Moreover, the processing and memory capabilities of the processing resource can be published and made available within the GUI application. In this way, and enterprise can monitor and alter network traffic as needed with the teachings of the present invention, without the need to acquire additional hardware and software resources.

[0060] Although specific embodiments have been illustrated and described herein, it will be appreciated by one of ordinary skill in the art that any arrangement that is calculated to achieve the same purpose may be substituted for the specific embodiments shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is intended that this invention be limited only by the claims and the equivalents thereof.